TC-LOC-201

HIRDLS

HIGH RESOLUTION DYNAMICS LIMB SOUNDER

Originator: Gary Heyman

Date: 1997-04-04

Subject / Title: FIR Filter Timing Test

This is a report on a timing test run on an R6000 engineering model computer to better estimate the processing load on the IPU processor due to FIR filter processing on the radiometric and ancillary data.

Key Words: timing, test, processing, demodulation, FIR, filtering, flight software

Reviewed By: Approved By:

Advanced Technology Center

Lockheed Martin Missiles & Space

3251 Hanover Street

Palo Alto, CA 94304-1191

USA

1 Scope

This TC presents the results of a timing test for the FIR filter algorithm performed on an engineering model R6000 VME-based computer while attending the RAD6000 training session at Lockheed Martin Federal Systems Division on March 25 - 27, 1997.

2 Reference Documents

TC-UCB-005 HIRDLS In-Flight Signal Processing

3 Signal Processing Description

While attending the R6000 training a program (firtst.c) was coded and run which performs data demodulation and a 32 tap FIR filter on one chopper revolution of data. The code was written based on infomation found in TC-UCB-005 (HIRDLS In-Flight Signal Processing) and is shown in Appendix A. The program uses arrays of 13 (16 bit integers) which were demodulated into 6 (64 bit floating point) "samples" for each of 32 detectors followed by a 32 tap FIR filter with dummy FIR coefficients. A total of 32 detectors were used (instead of 21) to accomodate possible demodulation and FIR filter processing of HIRDLS pointing data.

Test Setup: A VME based R6000 running at 5 MhZ was the target system. The VxWorks kernel was running and a VxWorks command shell was used to download and run the program from a Sun Workstation. We used the VxWorks function call "timexN(firtst)" to execute and calculate the execution time of the FIR test algorithm "firtst".

Results: The Execution time was 9.53 msec @ 5MhZ which translates to:

2.38 msec @ 20 MhZ

or

1.44 msec @ 33 MhZ

The nominal time per chopper revolution is 12.0 msec (83.3 Hz) so the CPU usage is:

2.38/12.0 = 19.83% @ 20 MhZ

or

1.44/12.0 = 12.00% @ 33 MhZ

These results are consistent with preivous estimates and represent a signifcant load to the IPU processor but should allow completion of the FIR processing to be accomplished in the IPU rather than requiring a separate processor (DSP) for this task.

Notes: The "firtst” program is very simplistic. It assumes the input data resides in memory (i.e. no I/O is required to get it) and no validity checking is done on the results. Nor is any telemetery formatting included. If the demodulation coefficients are indeed ±1/2 and ±1 then some time might be saved by not using floating point operations for the demodulation.

4 Acronyms

FIR	Finite duration Impulse Response
VME	Versabus Module - European

--END--

Appendix A FIR Timing Program

```
/* Gary Heyman           fir.c           3/21/97

 * Project: HIRDLS

 * Subsystem: IPU

 *

 * Description: Test program for SPU Data decomutation and FIR filter

 *

 */

#include <stdio.h>


/* <<<<<<<<<<<<<<<<<<<< PRIVATE DEFINES >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> */

#define D1 -0.5

#define D2 +1.0

#define D3 -0.5

#define NUM_DETECTORS 32

#define NUM_FIR_TAPS 32

#define SAMPLES_PER_REV 12


/* <<<<<<<<<<<<<<<<<<<<< private variables >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> */


/* <<<<<<<<<<<<<<<<<<<<<< FUNCTION PROTOTYPES >>>>>>>>>>>>>>>>>>>>>>>>>>>> */

void firtst(int reset);


int main(void)

{
```

```
  firtst(1);
  while(1) {
    firtst(0);
  }
/*  return (0);    */
}

void firtst(int reset)
{

  static int sample[SAMPLES_PER_REV+1][NUM_DETECTORS];
  static double decom[NUM_FIR_TAPS][NUM_DETECTORS], q[NUM_DETECTORS];
  int det,s,d,tap;
  static double C[NUM_FIR_TAPS]={1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9,1.10,
                 1.11,1.12,1.13,1.14,1.15,1.16,1.17,1.18,1.19,1.20,
                 1.21,1.22,1.23,1.24,1.25,1.26,1.27,1.28,1.29,1.30,
                 1.31,1.32};
  static double D[3]={D1,D2,D3};

  /* initialize arrays */
  if (reset) {
    for (s=0; s<(SAMPLES_PER_REV+1); s++)
        for (det=0; det<NUM_DETECTORS; det++)
          sample[s][det]=0;
    for(tap=0; tap<NUM_FIR_TAPS; tap++)
        for (det=0; det<NUM_DETECTORS;det++)
```

```
            decom[tap][det]=0;
      for (det=0; det<NUM_DETECTORS; det++)
            q[det]=0;
    }

    /* decom the s array into d */
    for (d=0; d<6; d++) {
      for(det=0; det<NUM_DETECTORS; det++) {
            s=d<<1;
            decom[d][det]= sample[s][det]*D[0]
                          +sample[s+1][det]*D[1]
                          +sample[s+2][det]*D[2];
      }
    }

    /* 32 Tap FIR */
    for (tap=0; tap<NUM_FIR_TAPS; tap++) {
      for (det=0; det<NUM_DETECTORS; det++) {
            q[det]+= C[tap]*decom[tap][det];
      }
    }
}
```